

A PSPACE-complete Graph Nim

Kyle G. Burke

Olivia C. George

Wittenberg University

oliviageorge@yahoo.com

kburke@wittenberg.edu

August 31, 2011

Abstract

We build off the game, NIMG [7] to create a version named NEIGHBORING NIM. By reducing from GEOGRAPHY, we show that this game is PSPACE-hard. The games created by the reduction share strong similarities with UNDIRECTED (VERTEX) GEOGRAPHY and regular NIM, both of which are solvable in polynomial-time. We show how to construct PSPACE-complete versions with nim heaps *1 and *2. This application of graphs can be used as a form of game sum with any games, not only NIM.

1 Background

1.1 Combinatorial Games

Combinatorial games are perfect-information games played between two players. The game proceeds by alternating turns where each player makes one move. Under normal play, the last player to make a move wins (if you can't make a move, you lose the game). A game ruleset is the description

of the moves (“options”) available to each player from a given position. All rulesets studied in this paper are *impartial*, meaning that at any state, both players always have the same move options. Thus, the identities of the players do not matter as far as strategies are concerned. With impartial game rules, game states (or “positions”) fall into one of two categories: P and N. P, or “zero”, is the class of positions where the previous player has a winning strategy (the next player then does not). N, or “fuzzy” games are the opposite: the next player does have a winning strategy from positions in this class.

1.2 Computational Complexity of Games

The “success” of a combinatorial game ruleset is partially dependent on the difficulty of the game: given a position, how hard is it to determine which player has a winning strategy, i.e., is that position in a P or N position? Luckily, we can use a tool in computer science to help answer this question. Computational complexity is a worst-case description of the amount of time (or other resources) required for an algorithm to solve a problem. Some problems do not have a known “efficient” algorithm and many of these are categorized into a hierarchy of partially-ordered complexity classes (P, NP, PSPACE, EXPTIME, etc.). For example, we know that the hardest problems in PSPACE (known as PSPACE-complete) are at least as difficult as the hardest in NP (NP-complete). Although it is not known whether problems in these classes can be efficiently solved, at the time of this writing no such algorithm has been discovered, and these classes provide a suitable metric for determining the difficulty of evaluating the outcome class of a game position.

In this paper, we show that NEIGHBORING NIM is a PSPACE-hard rule-set, meaning it is at least as hard to solve as any problem in PSPACE. To do this, we will show that any other problem in PSPACE can be rewritten as an equivalent instance of NEIGHBORING NIM. Thus, if an efficient solution to NEIGHBORING NIM exists, that algorithm can be used to solve any other problem in PSPACE. Since there are other known PSPACE-hard problems, it is sufficient to show a reduction from only one of those. This reduction is described in section 4.

All of the assertions above forgive a polynomial quantity. To learn more about this and algorithmic combinatorial game theory, see [?].

1.3 Nim

NIM is an impartial game played on a collection of piles (“heaps”), each with a non-negative number of “sticks”. On a player’s turn, they choose a non-empty pile and remove as many sticks as desired (at least one) from that one pile. A player loses when they have no non-empty piles to choose from.

The game of NIM is the classic impartial game, being the basis of Nimbers and Sprague-Grundy theory[6][4]. NIM has a simple definition and lots of nice properties, from easy evaluation of games to obvious composition of two NIM games (the sum is just a new NIM game).

1.4 NimG

NIM has been extended to incorporate graphs so that nim heaps are assigned to either edges or vertices. There are three different versions of the game named NIMG. In all three versions, a turn consists of both traversing an

edge of the graph and removing sticks from a visited element.

Fukuyama describes NIMG where nim heaps are embedded into the edges of the graph[3]. On each turn, the current player chooses an edge to traverse (which has at least 1 stick on it) and removes any number of sticks from that edge. The next player then starts on the vertex on the other end of that edge and must choose an adjacent edge for their move. When there are no more edges with sticks adjacent to the current vertex, the current player loses. This game continues to be studied, with recent results on complete graphs[1].

In 2004, Stockman studied VERTEX NIMG as part of an REU[7]. Here players similarly move from one vertex to another, but heaps are connected to the vertices instead of edges. The two variants can be easily described here as: “remove sticks, then move” and “move, then remove sticks”. In both cases, a player loses if they cannot complete their turn.

2 Notation

We begin by presenting some notation used throughout the paper.

A game *position* is a description of a game state.

Rigorously speaking, a *ruleset* is a function from positions to sets of positions, describing the move options for either player. Since all rulesets studied here are impartial, the range of all rulesets will be the power set of positions.

Often, the word, *game*, is overloaded to refer either to a position or a ruleset. This happens in this work as well, though the context should clarify the meaning.

We use the word *sticks* to refer to the objects in nim heaps. Thus, a nim

heap of size six contains six sticks.

An *optimal sequence set* is a set of sequences of plays for both players such that any move deviating from one of the sequences results in an N-position. No move in that sequence should be non-optimal for either player. Thus, if a player does not know whether they have a winning strategy, adhering to an optimal sequence is at least as good as any other move.

3 Neighboring Nim

We define the ruleset NEIGHBORING NIM to be similar to the “move then remove” version of NIMG, but also allow players to choose to play on the same vertex as the last move as though each vertex has a self-loop. Note that standard NIM is equivalent to a game of NEIGHBORING NIM on a complete graph with each heap on a separate vertex.

Definition 3.1 (Neighboring Nim)

Given an undirected graph, $G = (V, E)$ and weights on the vertices $w : V \rightarrow \mathbb{N}$, NEIGHBORING NIM is the ruleset where a move is made by choosing a vertex, v , where $w(v) > 0$ and reducing the weight of that vertex. If there was a previous move (in all but the first move) at vertex v' , then v must be chosen so that either $v = v'$ or v neighbors v' . (Meaning, (v, v') or $(v', v) \in E$).

Our main result for this paper is that this game is PSPACE-hard. Since our analysis uses graphs with a small number of sticks on each vertex, we define a version of the game with a bounded number of sticks per vertex.

Definition 3.2 (k -Neighboring Nim)

k -NEIGHBORING NIM is the same ruleset as NEIGHBORING NIM, except that the weight function w has bounded range: $[0, k]$.

We are able to show that 2-NEIGHBORING NIM is PSPACE-complete, and thus c -NEIGHBORING NIM is also PSPACE-complete for any constant $c \geq 2$. The case for 1-NEIGHBORING NIM is solvable in polynomial time, since this game is equivalent to UNDIRECTED (VERTEX) GEOGRAPHY [2]. Thus, if $P \neq PSPACE$, allowing a second stick on some vertex-heaps can greatly increase the computational hardness of determining the winning player.

4 Hardness of Neighboring Nim

Consider the problem of deciding who can win a game of NEIGHBORING NIM:

Definition 4.1 (NEIGHBORING – NIM)

NEIGHBORING – NIM is the problem of determining whether the current player has a winning strategy in a given game of NEIGHBORING NIM. This is equivalent to determining whether the current position is in N or P .

The following is the main result of this paper.

Theorem 4.2 (Hardness) NEIGHBORING – NIM is PSPACE-hard.

We will show the hardness of this problem by reducing from the game problem GEOGRAPHY, which is PSPACE-hard[5].

Proof. Given any GEOGRAPHY position, we will give an algorithm to construct an equivalent NEIGHBORING NIM state, meaning that there is a win

in the GEOGRAPHY position exactly when there is a win in corresponding NEIGHBORING NIM position. First we will describe the method for generating these positions, then prove their equivalence.

Let GG be a GEOGRAPHY position on the directed and unweighted graph $G = (V, E)$. We define a new undirected graph, $G' = (V', E')$ with weights on the vertices $w : V' \rightarrow \mathbb{N}$ in the following way. $\forall v \in V$: let $X_v \in V'$ and set $w(X) = 1$. Also, $\forall (y, z) \in E$: (edge directed from y to z) let $a_{y,z}, b_{y,z}, c_{y,z}, d_{y,z}, e_{y,z}, f_{y,z}, g_{y,z} \in V'$ where, ignoring the (y, z) -subscripts, $w(a) = w(b) = w(c) = w(e) = w(f) = w(g) = 1$, $w(d) = 2$ and,

$$(X_y, a), (a, b), (b, c), (c, d), (b, e), (e, f), (d, f), (d, g), (f, g), (g, X_z) \in E'.$$

See Figure 1 for a visual description.

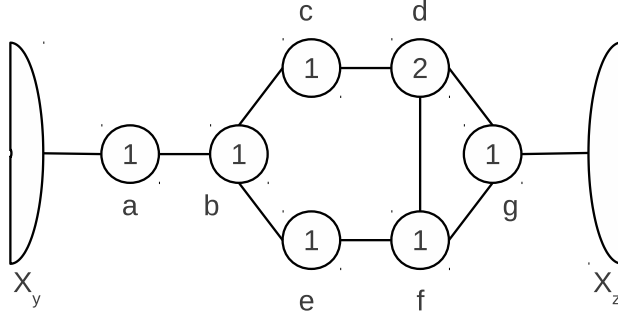


Figure 1: Our main gadget: reduce each directed edge from y to z to the undirected weighted graph shown here.

The resulting G' is the graph for our NEIGHBORING NIM position equivalent to GG . The only final step is to declare that if GG has a starting vertex, v , then $X_v \in V'$ is the starting vertex (where the previous play had been made) in our game and $w(X_v)$ is set to 0 instead of 1.

To complete the reduction, we must show that the structure in Figure 1 “acts” like a directed edge in GEOGRAPHY. Thus, we must prove:

- Moving “backwards” is a losing play. If the previous play was at X_z , then a backwards play would be to remove the only stick at $g_{(y,z)}$. A backwards play results in an \mathcal{N} -position.
- The same player moving into the gadget should also move out. If a player moves from X_y to $a_{(y,z)}$, then in an optimal sequence of plays, the same player will move from $g_{(y,z)}$ to X_z .

We prove the former in Lemma 4.3 and the latter in Lemma 4.4. The result is that each of these gadgets (as in Figure 1) in the NEIGHBORING NIM position works just like a (directed) edge in GEOGRAPHY. Trying to go backwards will result in losing and, if players play optimally, they both might as well continue through each gadget.

Lemma 4.3 (Don’t Go Backwards) *Any play from (X_z) to $g_{(y,z)}$ (for all y) results in an \mathcal{N} -position.*

(See Appendix A for a proof of this claim.) This implies that our gadgets are directed: if a player tries to go “backwards”: from an X -vertex to an i -vertex, the opponent will have a winning strategy.

To finish showing that our gadget acts like a directed edge, we must prove that “nothing can go wrong” during a regular forward traversal of the structure. To this end, we find two sequences that constitute an optimal sequence set through the gadget, thus showing that neither player benefits from deviating from the sequence. In order to get from one end of the gadget (as in Figure 1) to the other, the following sequence of moves suffices (let

Alice and Bob be our two players; we will again ignore subscripts): Alice “takes” a , Bob takes b , Alice takes e , Bob takes f , Alice decrements d by 1, Bob takes g , Alice takes X_z . Note that the same player (in this example, Alice) who chooses to take a also moves to X_z . The other sequence is where Bob takes c instead of g —here Alice will take the remaining object at d and Bob will be forced to take g , rejoining with the first sequence. See Figure 2 for a visual description of the safe sequences. We must prove that neither player benefits from deviating from these sequences. To do this, we show that any deviation is a losing move.

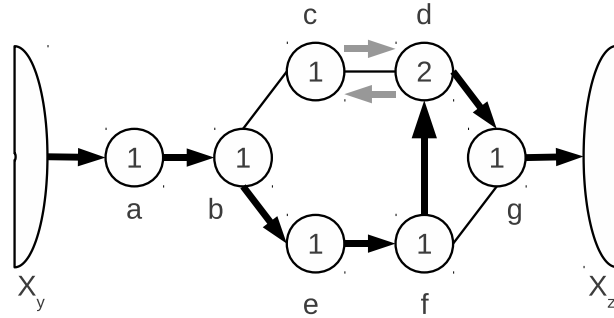


Figure 2: This sequence of moves is “safe” for both players to traverse the gadget. The two gray arrows indicate the potential additional moves in the second sequence. Each move assumes that exactly one object is taken from a vertex.

Lemma 4.4 (Stick to the Script) *Let the notation $k(p)$ denote taking p objects from vertex k in a turn. Then, after the plays $(\dots, X_y(1), a(1))$ any play deviating from the following sequences is a losing move:*

$(b(1), e(1), f(1), d(1), g(1), X_z(1))$

$(b(1), e(1), f(1), d(1), c(1), d(1), g(1), X_z(1)).$

(See Appendix B for the proof of this claim.) This implies that once a player makes an appropriate move onto the gadget (playing on an a -node) any “safe” sequence of moves in the gadget results in that same player making the play at the opposite X node. The two above claims combined show that our gadget correctly models a directed edge in a graph just between the X nodes.

Thus, for any edge (y, z) in our GEOGRAPHY position, GG , the move to $a_{(y,z)}$ will result in the same player moving to X_z as desired. Also, since we proved players shouldn’t go backwards, this game is equivalent to GG ; the first player has a winning strategy in GG exactly when the first player has a winning strategy in this NEIGHBORING NIM position.

Thus, NEIGHBORING – NIM is PSPACE-hard. \square

Unfortunately, NEIGHBORING – NIM is not automatically PSPACE-complete as games could have a number of moves exponential in the size of the description of the game. For example, a vertex can have a number of sticks exponential in the amount of bits needed to express that number and the rest of the graph. We leave this as an open problem.

Open Problem 4.5 *Is NEIGHBORING – NIM in PSPACE?*

We can sidestep this problem a bit by using our bounded-sticks version of the game. First, let’s define the problem.

Definition 4.6 (k – NEIGHBORING – NIM)

k – NEIGHBORING – NIM *is the problem of determining whether the current player has a winning strategy in a game of k -NEIGHBORING NIM.*

For any constant k , k – NEIGHBORING – NIM is in PSPACE. We note that our reduction for general NEIGHBORING NIM has a maximum weight of

2, and thus works for any game where $k \geq 2$. This leads to our completeness result.

Corollary 4.7 (2-Neighboring Nim Completeness)

k – NEIGHBORING – NIM is PSPACE-complete for any $k \geq 2$.

5 Generalization and Conclusions

Building on algorithmic work analyzing different versions on NIMG, we present NEIGHBORING NIM, a new PSPACE-hard game.

An interesting aspect of the hardness of NEIGHBORING NIM is the juxtaposition with VERTEX GEOGRAPHY. 1-NEIGHBORING NIM is the same ruleset as UNDIRECTED VERTEX GEOGRAPHY, which is solvable efficiently[2]. However, by adding an extra stick to just a few vertices, we can push the game into PSPACE-hardness!

Additionally, we see that we can use this graph-embedding technique with games other than NIM. Given a graph, assign different game states to the vertices, and use similar rules: players may make one move legal in the game in any vertex neighboring the last play. For any game which has instances equivalent to *2 and *1, the “Neighboring”-version of that game is PSPACE hard, using an analogous construction to the one specific to NIM. With this, we have our main result.

Theorem 5.1 (General) *For any two game positions, G_1 and G_2 , with values *1 and *2, respectively, we can define a new ruleset, NEIGHBORING- G_1, G_2 , which is PSPACE-complete.*

Note that we are not restricted to adding instances of the same game. This is a general method for adding games together; the standard (disjunc-

tive) sum of n games here is equivalent to the “Neighboring”-version played on the complete graph with n vertices.

6 Acknowledgments

The authors would like to thank those who played NEIGHBORING NIM with them, including three excellent Wittenberg students: Deanna Fink, Dang Mai and Ernie Heyder, as well as Professor Doug Andrews who defeated the first author over and over again. We would also like to thank Adam Parker for listening to initial versions of the hardness proof as well as proofreading this paper.

References

- [1] Lindsay Erickson. Nim on Complete Graphs. *Ars Combinatoria*, Accepted for Publication in 2009.
- [2] Aviezri S. Fraenkel, Edward R. Scheinerman, and Daniel Ullman. Undirected edge geography. *Theor. Comput. Sci.*, 112(2):371–381, 1993.
- [3] Masahiko Fukuyama. A nim game played on graphs. *Theor. Comput. Sci.*, 1-3(304):387–399, 2003.
- [4] P. M. Grundy. Mathematics and games. *Eureka*, 2:198—211, 1939.
- [5] David Lichtenstein and Michael Sipser. Go is polynomial-space hard. *J. ACM*, 27(2):393–401, 1980.
- [6] R. P. Sprague. Über mathematische Kampfspiele. *Tôhoku Mathematical Journal*, 41:438—444, 1935-36.

[7] Gwendolyn Stockman. Presentation: The game of nim on graphs: NimG, 2004. Available at http://www.aladdin.cs.cmu.edu/reu/mini_probes/papers/final_stockman.ppt.

A Proof of Lemma 4.3

Lemma A.1 (Don’t Go Backwards) *Any play from (X_z) to $g_{(y,z)}$ (for all y) is suboptimal.*

We will refer to the player who moves from X_z to g (we will leave out the subscript for the internal vertices) as the “foe” while the other player is the “hero”. We will show that the hero has a winning strategy after a backwards move. We can now look at two cases, each depending on the state of the game outside the gadget.

The first is the case where the move from a to X_y would be a winning play. In this case, the hero can next move from g to d and take both of the objects there. The foe has two options, both of which, we show, allow the hero to win.

1. **The foe moves to c .** In this case the hero must choose to go to b . The foe can now either choose to move to a —in which case the hero will gladly move to X_y and win as we assumed—or to e . Then the hero simply takes the object at f and, as there are no more moves, the hero has won.
2. **The foe moves to f .** The hero must then take e and the foe must take b . The hero can then move to c and win the game.

The second major case assumes that the move from a to X_y is a losing play. Here, the hero can still move to d (from g) but will take only one of

the objects. Now the foe has three options: taking the other object at d , moving to c or moving to f . We show all to be losses.

1. **Foe moves to c .** Now the hero should take the remaining object at d . The following sequence must occur: foe must take f , hero at e , foe at b , hero at a , followed by the foe at X_y , a losing move by our assumption.
2. **Foe takes the remaining object at d .** The hero will choose to take c , so the foe must take b . The hero can then take a , forcing the foe to take X_y , a losing move by our assumption.
3. **Foe takes f .** The hero should then take e so the foe must take b . Again, the hero can take a , so the foe must move to X_y , a losing move by our assumption.

Thus, it is a losing play to move from an X -vertex to a g -vertex.

B Proof of Lemma 4.4

Lemma B.1 (Stick to the Script) *Let the notation $k(p)$ denote taking p objects from vertex k in a turn. Then, after the plays $(\dots, X_y(1), a(1))$ any play deviating from the following sequences is a losing move:*

$$(b(1), e(1), f(1), d(1), g(1), X_z(1))$$

$$(b(1), e(1), f(1), d(1), c(1), d(1), g(1), X_z(1)).$$

We continue by analyzing all possible deviations from these sequences and show that they result in a loss. In this claim, we will refer to the deviating player as the foe and the other player as the hero. We will show

that the foe loses in each case. It may be helpful to refer to Figure 2 during these case descriptions.

1. **$c(1)$ instead of $e(1)$.** Here we have two subcases: either moving from g to X_z is a winning (result is a \mathcal{P} -position) or losing (a \mathcal{N} -position) move. If it's in \mathcal{N} , then the hero can respond to $c(1)$ with $d(1)$. If the foe then chooses $g(1)$, the hero can take the remaining stick in d with $d(1)$. $f(1)$ and $e(1)$ must follow with the hero winning. If the foe instead chooses $f(1)$, the hero can win instantly by choosing $e(1)$. For the foe's last chance, they could select $d(1)$, removing the other stick from d . The hero should respond with $g(1)$. The foe will lose by selecting $f(1)$, because the hero will win at $e(1)$, but the foe will also lose with $X_z(1)$, an \mathcal{N} -position as assumed.

If $X_z(1)$ is instead leaves the board in \mathcal{P} , the hero should respond to $c(1)$ with $d(2)$. The foe could choose $f(1)$, but the hero can then win with $e(1)$. Instead, the foe can choose $g(1)$ in which case the hero can choose $X_z(1)$ and win, as assumed.

2. **$d(2)$ instead of (the first) $d(1)$.** Here the hero has a simple move to win. By taking $c(1)$ there are no further moves and the foe has lost.
3. **$g(1)$ instead of (the first) $d(1)$.** The hero can respond with $d(1)$. This leaves two different adjacent vertices with 1 object apiece and no other adjacent non-empty vertices. Either move by the foe results in one remaining move and a win for the hero.
4. **$d(1)$ instead of $c(1)$.** The hero can respond with $c(1)$ and win.
5. **$d(1)$ instead of $X_z(1)$.** This cannot happen in the second sequence, but if it happens in the first, the hero can respond with $c(1)$ and win.

Thus, any deviation from the two sequences specified in the claim puts the game in an \mathcal{N} -position.